

AMENDMENT

In the Claims

Please amend claims 1-3, 7-9, 11, 15, and 17.

Please cancel claims 4-6, 10, 12-14, 16, and 18-20 without prejudice.

Please add claims 21-37.

1. (Currently Amended) A method comprising:
implementing a firmware-based virtual machine monitor (VMM) upon a computing system having a native environment that executes in physical mode; and
~~executing the virtual machine monitor in a most privileged mode, the virtual machine monitor emulating physical mode such that the native environment is executed in a less privileged mode~~ emulating legacy hardware components that are not present in the native environment using the VMM to provide support for legacy code that presupposes the existence of such hardware components.
2. (Currently Amended) The method of claim 1 wherein the native environment is ~~selected from the list including~~ comprises one of a 32-bit environment[[,]] or a 64-bit environment, ~~and a PC/AT environment.~~
3. (Currently Amended) The method of claim 2, wherein the VMM ~~contains code to provide functionality selected from the list consisting of~~ provides at least one of PC/AT hardware emulation[[,]] and PC/AT environment emulation, ~~secure storage, and secure execution.~~
4. (Cancelled)
5. (Cancelled)

6. (Cancelled)

7. (Currently Amended) A method comprising:

implementing a virtual machine monitor (VMM) on a computing system having an extensible firmware architecture that enables firmware to be provided from third parties;
and

executing ~~such that~~ untrusted firmware code via the VMM is executed in a
sandbox mode such that the code is prevented from harming the system during the pre-
boot phase of the computer system, wherein the code is given access to a subset of
system resources, while code access to other system resources is filtered by the VMM.

8. (Currently Amended) The method of claim 7, wherein the untrusted firmware
code is legacy BIOS code.

9. (Currently Amended) A machine-readable medium that provides executable
firmware instructions which, when executed by a processor, cause the processor to
perform ~~a method, the method~~ operations comprising:

implementing a firmware-based virtual machine monitor (VMM) upon a
computing system having a native environment that executes in physical mode; and
~~executing the virtual machine monitor in a most privileged mode, the virtual~~
~~machine monitor emulating physical mode such that the native environment is executed~~
~~in a less privileged mode~~ emulating legacy hardware components that are not present
in the native environment using the VMM to provide support for legacy code running on
the computer system.

10. (Cancelled)

11. (Currently Amended) The machine-readable medium of claim 10, wherein the VMM ~~contains code to provide functionality selected from the list consisting of~~ provides at least one of PC/AT hardware emulation[[,]] and PC/AT environment emulation, ~~secure storage, and secure execution.~~

12. (Cancelled)

13. (Cancelled)

14. (Cancelled)

15. (Currently Amended) An apparatus comprising:

a computing system having a native execution environment that executes in physical mode; and

a virtual machine monitor, ~~executed in a most privileged mode,~~ implemented thereon, the virtual machine monitor emulating ~~physical mode such that the native environment is executed in a less privileged mode.~~ legacy hardware components that are not present in the native environment to provide support for legacy code to run on the computer system.

16. (Cancelled)

17. (Currently Amended) The apparatus of claim 16, wherein the VMM ~~contains code to provide functionality selected from the list consisting of~~ provides at least one of PC/AT hardware emulation[[,]] and PC/AT environment emulation, ~~secure storage, and secure execution.~~

18. (Cancelled)

19. (Cancelled)

20. (Cancelled)

21. (New) The method of claim 1, further comprising:

loading the VMM during a pre-boot phase of the computer system having an extensible firmware framework, the VMM comprising a modular firmware component running on the extensible firmware framework.

22. (New) The method of claim 1, further comprising:

publishing an environment that appears to be a physical mode environment to software entities run on the VMM, wherein the physical mode environment includes a memory map that includes memory addresses below one megabyte, while the VMM is implemented on a computer system that does not decode physical addresses below one megabyte.

23. (New) The method of claim 2, further comprising:

enabling a legacy option ROM (read-only memory) to run and effect its input/output (I/O) services; and

translating the results of the I/O services into a native API (application program interface).

24. (New) The method of claim 7, further comprising:

trapping an attempted write access by a firmware program via the VMM;
determining if the firmware program is allowed access to a data structure to which the write access pertains; and, if so,

allowing the write access to be performed; otherwise,
denying the write access.

25. (New) The method of claim 24, wherein the operation of determining if the firmware program is allowed access to the data structure comprises:

determining if the firmware program is started by Extensible Firmware Interface (EFI) core code; and

determining if the firmware program is allowed access to EFI core data structures.

26. (New) The method of claim 1, wherein the VMM performs the further operation of hiding a portion of an address space for the computer system.

27. (New) A method, comprising:

implementing a virtual machine monitor (VMM) during the pre-boot phase of a computer system; and

authenticating an Extensible Firmware Interface (EFI) firmware module using the VMM.

28 (New) The method of claim 27, further comprising:

storing digital signatures of valid firmware modules in secure storage;

authenticating a firmware module via the VMM by comparing a digital signature provided with the firmware module to the digital signatures in the secure storage.

29 (New) The method of claim 28, further comprising:

maintaining an attestation log via the VMM identifying firmware modules that have been loaded and authenticated by the VMM.

30. (New) The machine-readable medium of claim 9, wherein the computer system includes an extensible firmware framework that enables third-party firmware modules to be loaded during the pre-boot phase of the computer system, and execution of the instructions cause further operations to be performed, comprising:

loading the VMM during the pre-boot phase of a computer system; and
authenticating a firmware module using the VMM.

31. (New) The machine-readable medium of claim 30, wherein execution of the instructions cause further operations to be performed, comprising:

authenticating a firmware module via the VMM by comparing a digital signature provided with the firmware module with digital signatures stored in secure storage that is accessible to the VMM.

32. (New) The machine-readable medium of claim 30, wherein execution of the instructions cause further operations to be performed, comprising:

maintaining an attestation log via the VMM identifying firmware modules that have been loaded and authenticated by the VMM.

33. (New) The machine-readable medium of claim 9, wherein the computer system includes an extensible firmware framework that enables third-party firmware modules to be loaded during the pre-boot phase of the computer system, and execution of the instructions cause further operations to be performed, comprising:

executing untrusted firmware module code on the VMM in a sandbox mode such that the code is prevented from harming the computer system.

34. (New) The machine-readable medium of claim 9, wherein execution of the instructions cause further operations to be performed, comprising:

enabling a legacy option ROM (read-only memory) to run and effect its input/output (I/O) services; and

translating the results of the I/O services into a native API (application program interface).

35. (New) The apparatus of claim 15, wherein the computer system includes an extensible firmware framework that enables third-party firmware modules to be loaded during the pre-boot phase of the computer system, and the VMM comprises a firmware component that is loaded during the pre-boot phase and is employed to authenticate firmware modules.

36. (New) The apparatus of claim 15, wherein the VMM authenticates a firmware module by comparing a digital signature provided with the firmware module with digital signatures stored in secure storage that is accessible to the VMM.

37. (New) The apparatus of claim 15, wherein the VMM performs further operations, including:

enabling a legacy option ROM (read-only memory) to run and effect its input/output (I/O) services; and

translating the results of the I/O services into a native API (application program interface).